

Taller Uno NetLogo

Parte Uno: Conceptos básicos

Introducción:

Netlogo es un entorno de programación que permite la simulación de fenómenos naturales y sociales. Fue creado por Uri Wilensky en 1999 y está en continuo desarrollo.

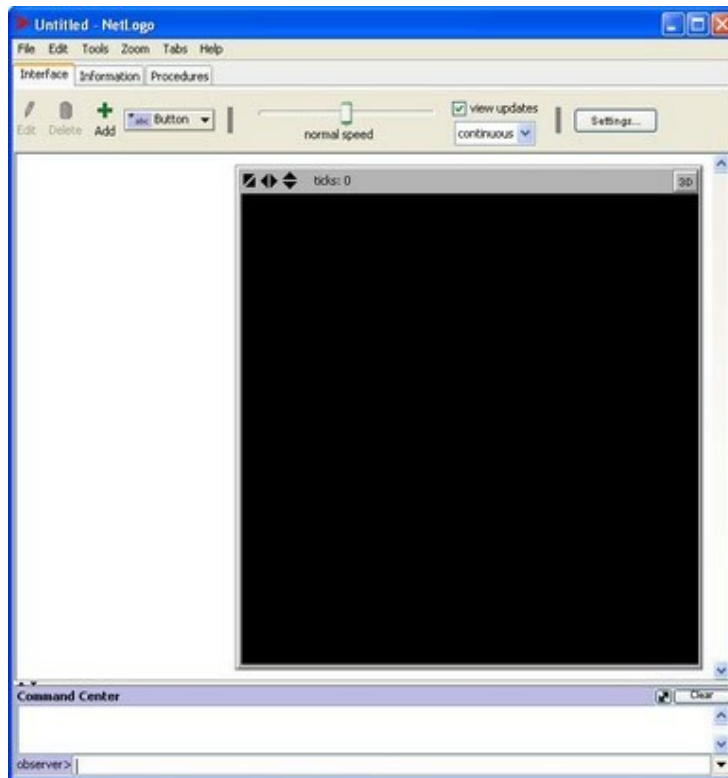
Netlogo es particularmente útil para modelar sistemas complejos que evolucionan en el tiempo. Los implementadores de modelos pueden dar instrucciones a cientos o miles de agentes para que todos ellos operen de manera independiente, entre sí y con el entorno. Esto hace posible explorar la relación entre el comportamiento a bajo nivel de los individuos y los patrones macroscópicos que surgen a partir de la interacción de muchos individuos entre sí .

Netlogo permite a los usuarios abrir simulaciones y “jugar” con ellas, así como explorar su comportamiento bajo una serie de condiciones. Asimismo, permite al usuario la creación de sus propios modelos. Netlogo es lo suficientemente sencillo como para que los estudiantes y los profesores puedan ejecutar las simulaciones o incluso construir las suyas propias. Además, su grado de desarrollo actual es suficiente como para servir como una herramienta potente para investigadores en muchos ámbitos.

Existe abundante documentación y tutoriales sobre Netlogo. El programa incluye una galería de modelos (*models library*), que contiene una amplia colección de simulaciones que pueden ser ejecutadas y modificadas. Este conjunto de modelos pertenece a ámbitos muy diversos, tanto de la naturaleza como de ciencias sociales (biología, medicina, física y química, matemáticas y computación, economía y psicología social).

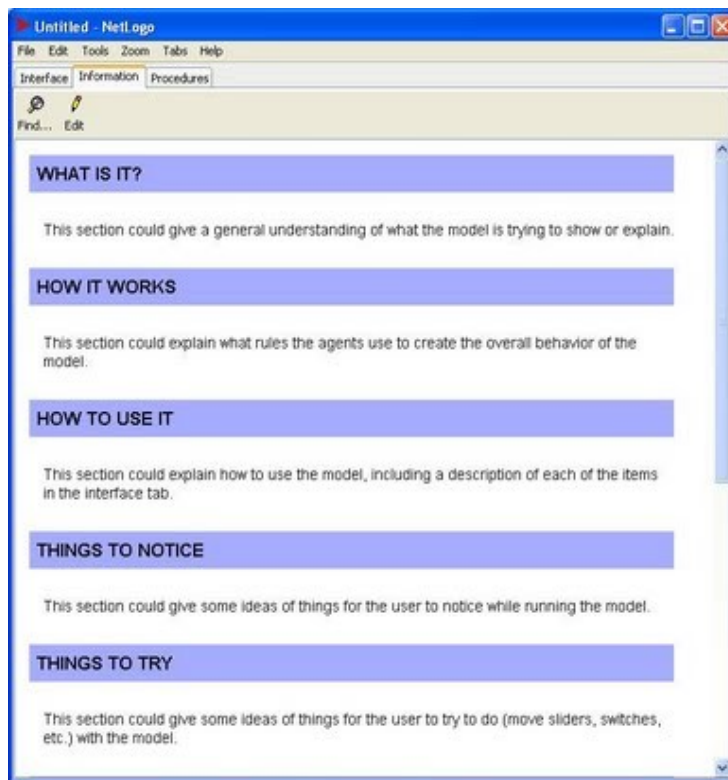
Interactuando Con NetLogo

Cuando arranquemos Netlogo, la pantalla de nuestro ordenador presentará el siguiente aspecto:

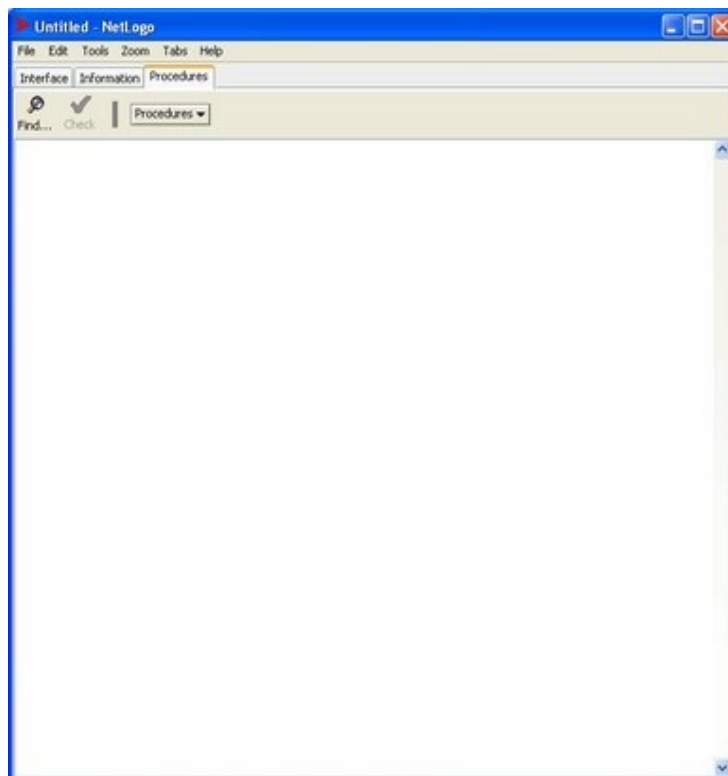


En la parte superior observamos tres pestañas: *interface* (interfaz), *information* (información) y *procedures* (procedimientos).

Aunque lo veremos con mayor detenimiento próximamente, en la primera de las pestañas (*interface*) será donde se represente nuestro modelo.
En la segunda pestaña (*information*) podremos añadir información sobre nuestro modelo para informar a los usuarios:

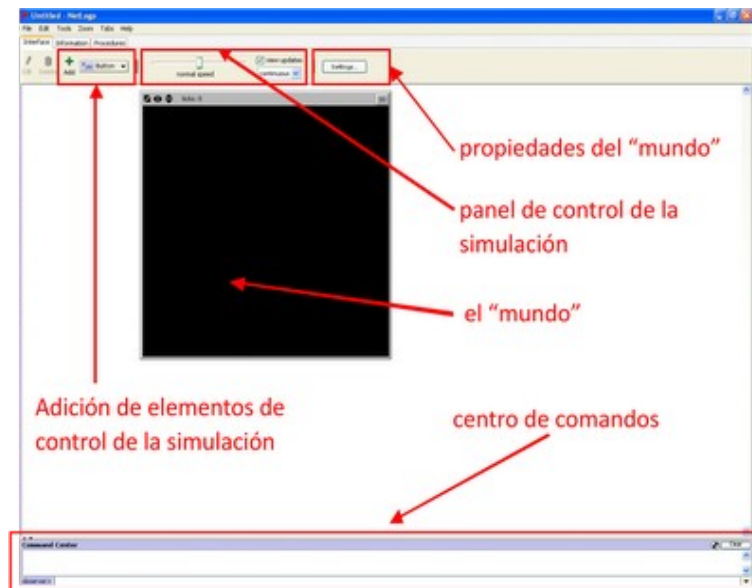


y en la última pestaña (*procedures*) escribiremos los procedimientos que se encargarán de llevar a cabo la ejecución de nuestro modelo:



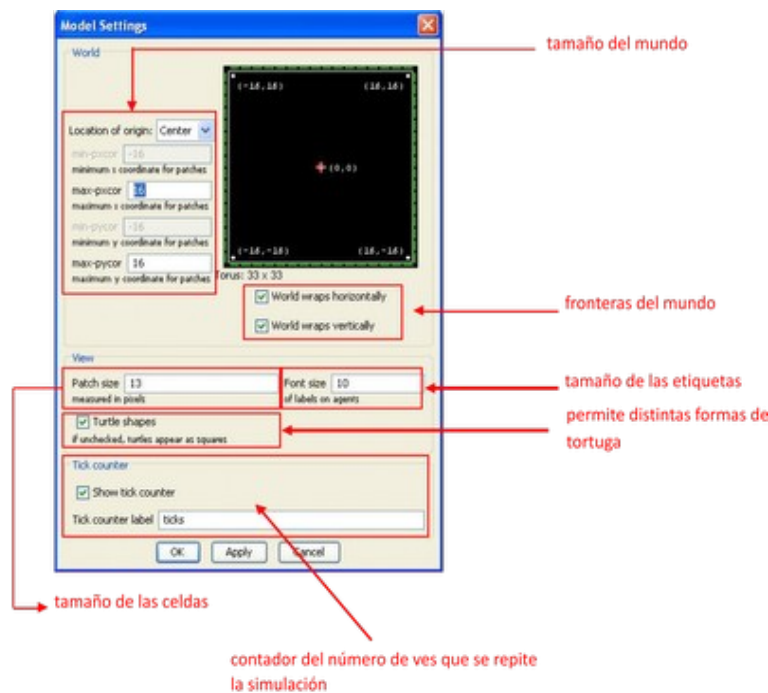
1. INTERFAZ (*interface*)

Dentro de la pestaña interfaz, distinguimos los siguientes elementos:



El "mundo" es el lugar donde se visualiza el comportamiento de los agentes que creamos en nuestro modelo. Está compuesto por una serie de *patches* o celdas. Se puede elegir el tamaño de cada celda y el número de celdas que hay en el mundo pulsando el botón "settings":

1.1. MODEL SETTINGS (AJUSTES DEL MODELO)



TAMAÑO DEL MUNDO

El tamaño del mundo queda definido por las variables **max-pxcor** y **max-pycor**, que representan la coordenadas centrales de las celdas más alejadas del origen.

¿Cómo sabemos entonces cuántas celdas tiene nuestro modelo de ancho?

Por defecto, **max-pxcor** tiene el valor 16 y el posición del mundo es centrada. Esto quiere decir que tenemos 16 celdas a la derecha del origen de coordenadas, y 16 celdas a la izquierda del mismo. A esto hay que añadirle la celda que está justo en el origen de coordenadas, por lo que el mundo tendrá $2 \cdot 16 + 1 = 33$ celdas de ancho.

Si **max-pycor** tiene el valor 16 como en la figura, de manera análoga, el mundo tendrá 33 celdas de altura.

En conclusión, el mundo tendrá unas dimensiones de 33x33 celdas.

Se cumple que las variables de Netlogo **world-width** y **world-height**, que almacenan la anchura y la altura del "mundo", cumplen las siguientes propiedades:

world-width = $2 * \text{max-pxcor} + 1$;; anchura del mundo

world-height = $2 * \text{max-pycor} + 1$;; altura del mundo

Además, se cumple que el tamaño del mundo (en número de celdas) será igual al producto de las variables **world-width** y **world-height**.

Puedes comprobarlo escribiendo lo siguiente en el centro de comandos:

show world-width ;; muestra la anchura del mundo (por defecto 33 celdas)

show world-height ;; muestra la altura del mundo (por defecto 33 celdas)

show count patches ;; cuenta el número de patches (celdas) y las muestra por pantalla

(por defecto 1.089)

FRONTERAS DEL MUNDO

Si la opción “*world wraps horizontally*” está activada, cuando una tortuga, al desplazarse, traspasa el límite del mundo, aparecerá por el lado opuesto del mundo. Si no está activada, no se permite que las tortugas avancen más allá de los límites del mundo, por lo que si esto ocurriera, recibiríamos un mensaje de error al ejecutar nuestro modelo.

Lo mismo es aplicable para la opción “*world wraps vertically*” en lo referente a los desplazamientos en vertical.

TAMAÑO DE LAS CELDAS (PATCHES)

Lo definimos en la ventana “*patch size*”, medido en pixels. Ten en cuenta que la anchura “visual” del mundo (es decir, el tamaño que va a tener el mundo en nuestra pantalla) dependerá tanto de la variable “*patch size*” como de *max-pxcor* y *max-pycor*: Cuanto mayor sea *max-pxcor*, más celdas tendremos en el eje x, y cuanto mayor sea el tamaño de cada celda, mayor será por tanto el tamaño visual del mundo.

TAMAÑO DE LAS ETIQUETAS DE LOS AGENTES

Lo introducimos en la ventana “*font size of labels on agents*”.

FORMA DE LAS TORTUGAS

Si la opción “*turtle shapes*” está desactivada, las tortugas se muestran como cuadrados. En cambio, si están activadas, podemos darles la forma que deseemos (en la barra de menús, elegir *tools* y a continuación *turtle shapes editor* para ver las formas de tortuga disponibles).

TICK COUNTER (CONTADOR DEL NÚMERO DE VECES QUE SE REPITE UNA SIMULACIÓN)

Para que este contador se incremente, debemos añadir la palabra “*tick*” dentro de nuestro código. Cuando la ejecución del programa pase por la palabra *tick*, este contador se incrementará en una unidad.

1.2. PANEL DE CONTROL DE LA SIMULACIÓN

VELOCIDAD DE SIMULACIÓN: Por defecto aparece velocidad “normal”. Puede variarse desplazando la barra hacia la derecha (más deprisa) o hacia la izquierda (más despacio). Ten en cuenta que, de hecho, el ordenador procesa los datos a la misma velocidad independientemente de lo elegido en esta barra, pues en realidad lo que se altera es la velocidad con la que estos datos se muestran en la pantalla.

VIEW UPDATES (CONTINUOUS / ON TICKS): Si esta casilla está activada, los cambios en el modelo se nos mostrarán “paso a paso”. En cambio, si está desactivada, el mundo sólo se actualizará al finalizar la simulación. Si se selecciona “*continuous*”, el mundo, variables y gráficas que se presentan en pantalla se actualizan paso a paso. En cambio, si se selecciona “*on ticks*”, la pantalla se actualiza al finalizar cada iteración (cada vez que aparezca la palabra *tick* en la ejecución del programa).

1.3. ADICIÓN DE ELEMENTOS DE CONTROL DE LA SIMULACIÓN

Se pueden incorporar a la interfaz:

BUTTON (botón). Al pulsarlo, se ejecutan una serie de comandos.

SLIDER (barra desplazadora). Controla el valor de una variable global. Permite acotar los valores de esta variable dentro de un rango, así como asignarle un valor por defecto.

SWITCH (interruptor). Controla el valor de una variable global booleana. Si está seleccionado “on” la variable tomará el valor true. En caso contrario, tomará el valor false.

CHOOSE (selector). Controla el valor de una variable global. A diferencia de la barra desplazadora (slider), los valores que podrá tomar esta variable serán discretos.

INPUT (entrada). Controla el valor de una variable global. El usuario teclea dicho valor.

MONITOR: Muestra el valor que toma una variable global durante la ejecución del programa.

PLOT (gráfico): Representación gráfica.

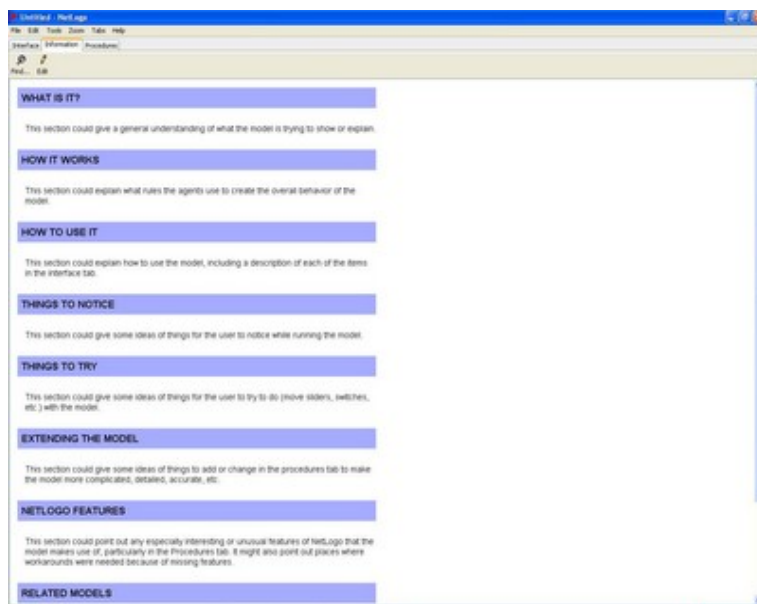
OUTPUT: (salida). Se trata de una pantalla en la que podemos mostrar mensajes a lo largo de la ejecución. Sólo se permite añadir un output por modelo.

NOTE (nota). Texto.

2. INFORMATION (*información*)

Muestra información sobre el modelo: qué es, cómo funciona, aspectos que observar, cuestionar que probar, modelos relacionados, etc.

Todo ello puede modificarse al pulsar el botón “edit”.



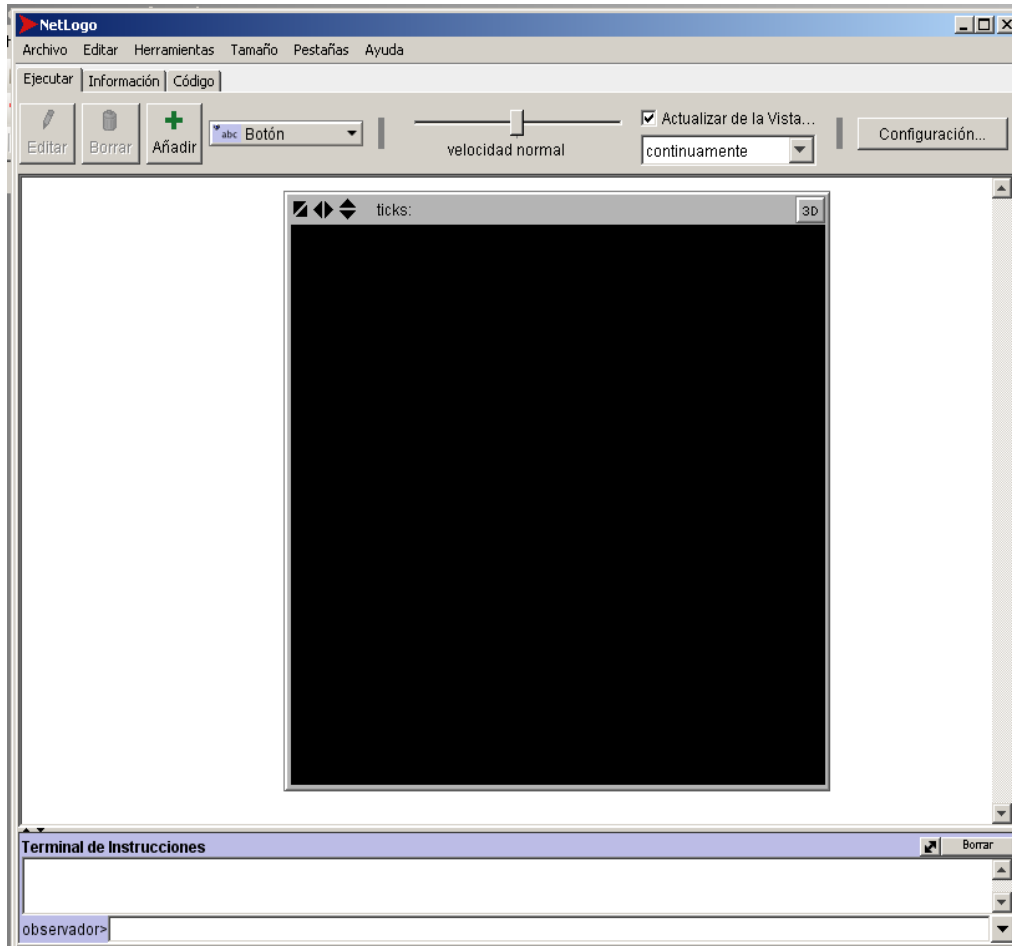
3. PROCEDURES (*procedimientos*)

En esta vista se escriben los procedimientos (“funciones”) necesarios para programar nuestro modelo.

Parte 2: Comandos básicos

La mejor manera de aprender a usar netLogo es sentarse al frente del computador y jugar un rato con los comandos básicos, comencemos:

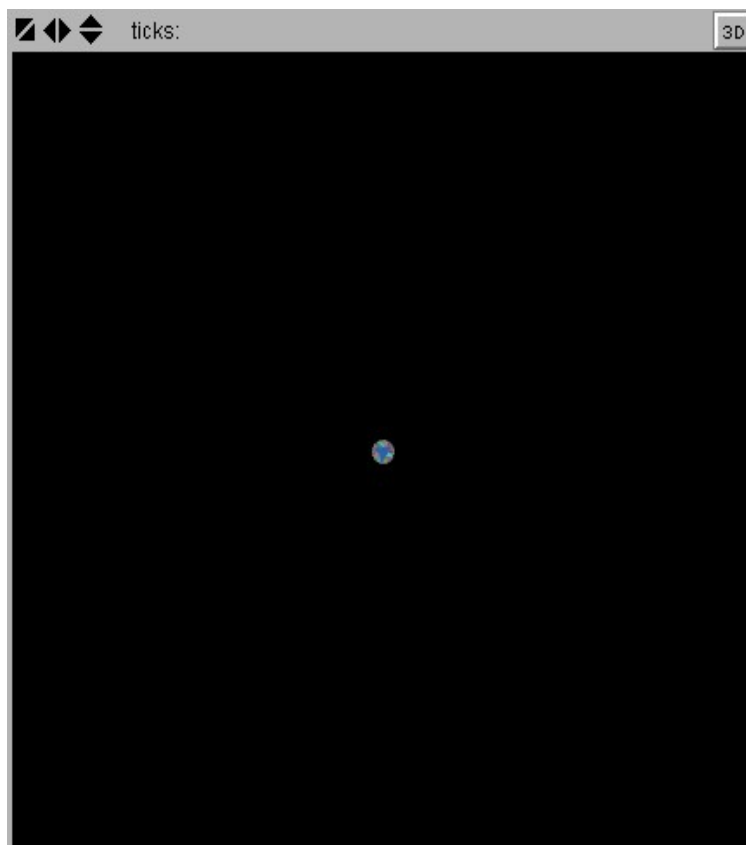
Arranquemos netLogo aparecerá la pantalla:



En la parte inferior de la pantalla, en la ventana llamada observador, teclee:

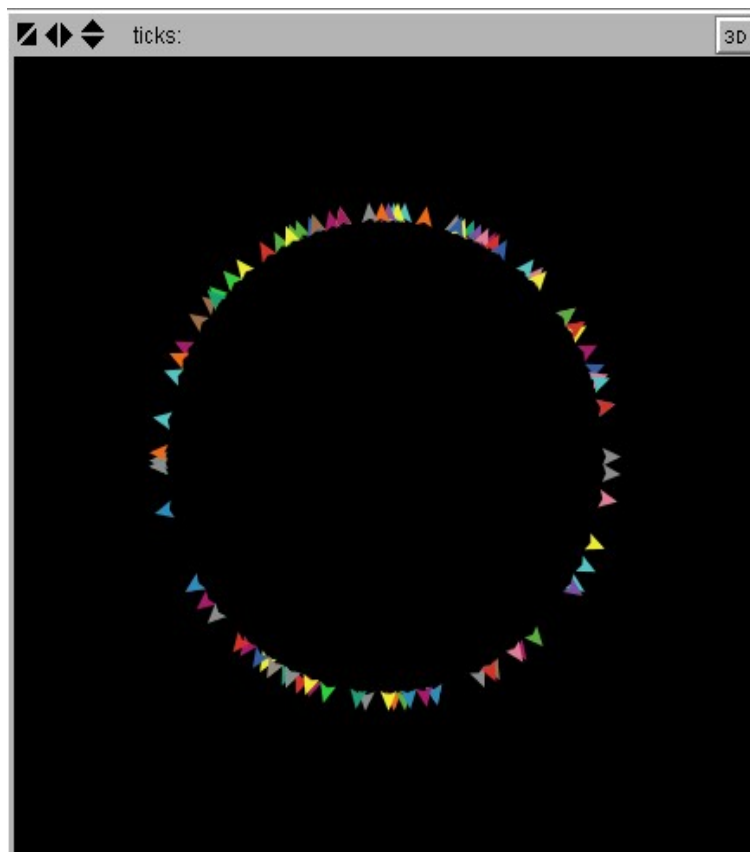
```
create-turtles 100
```

Aparece:



Las tortugas aparecen todas amontonadas en el centro de la pantalla, separémoslas un poquito, tecleemos:

```
ask turtles [ forward 10]
```

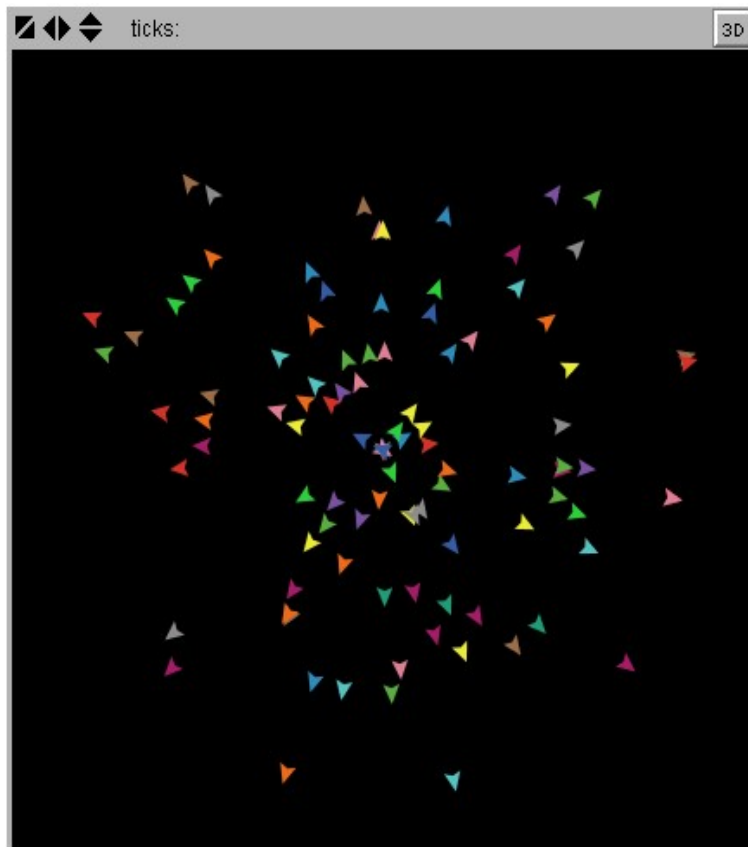


Devolvamos las tortugas a su posición inicial:

```
ask turtles [ back 10]
```

Ahora hagamos que las tortugas se muevan a diferentes distancias al azar (random)

```
ask turtles [ forward random 15]
```



Observe que cada tortuga se crea con una dirección al azar, de tal manera que hay tortugas en “todas” las direcciones.

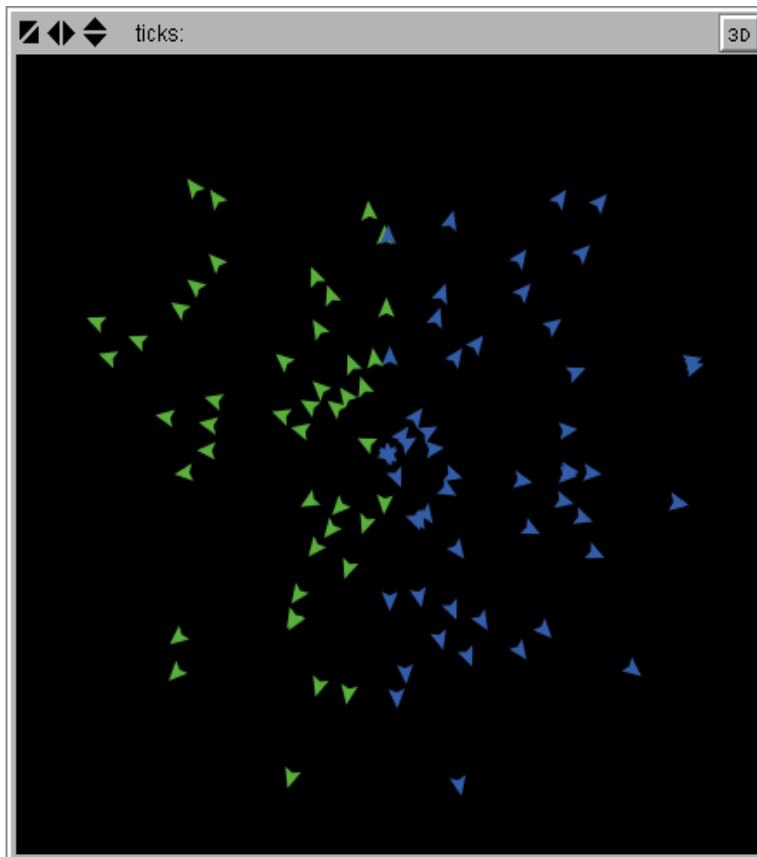
Cambiamos ahora de color a las tortugas:

```
ask turtles [set color blue]
```

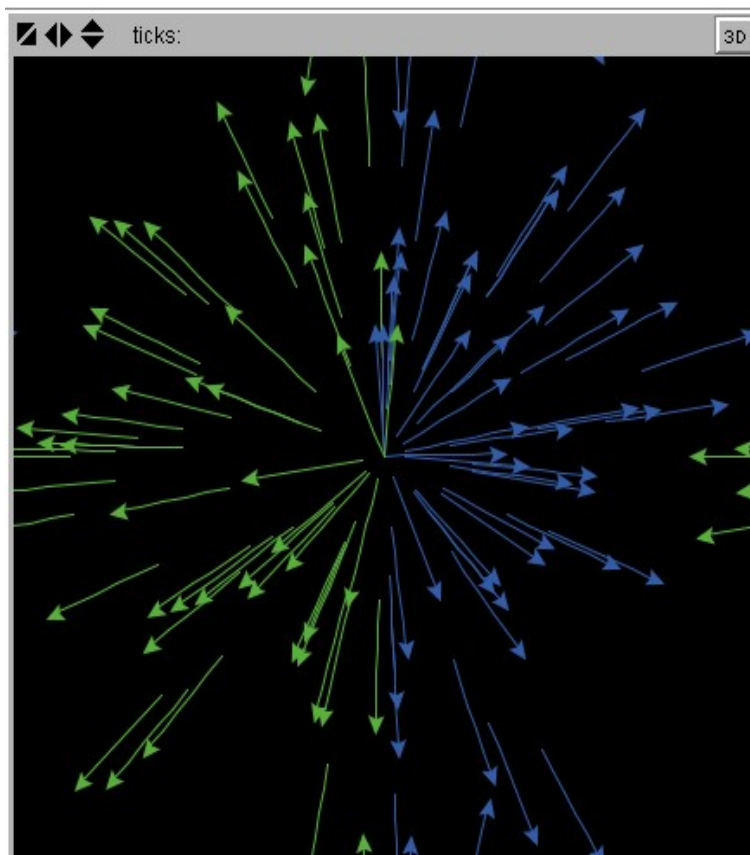
las tortugas se vuelven azules

```
ask turtles [if xcor < 0 [set color green] ]
```

en este caso las tortugas que tiene coordenada x negativa se convierten en verdes (el origen de coordenadas se encuentra en el centro de la pantalla)

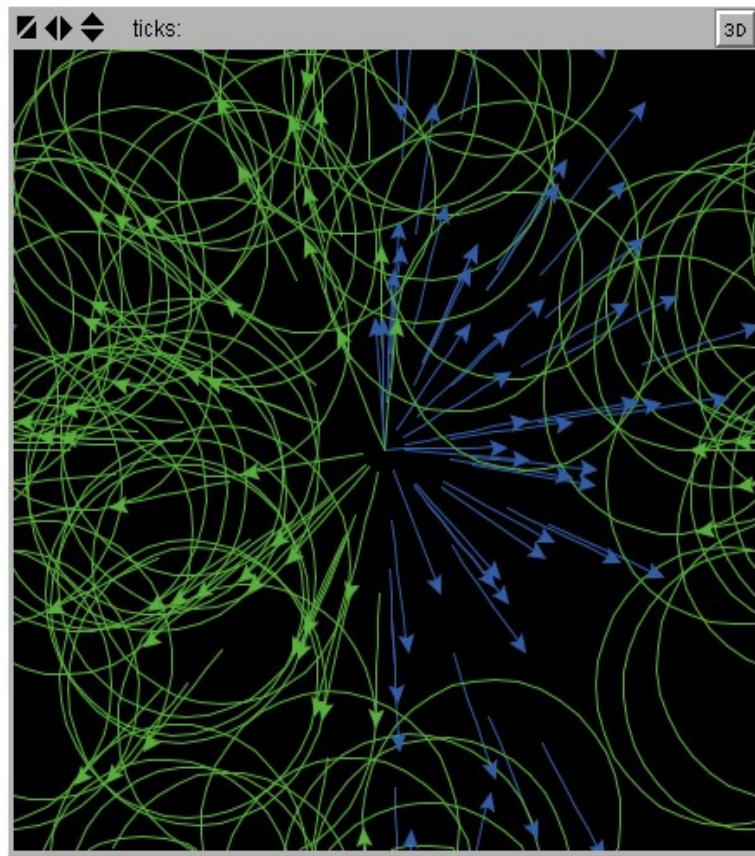


```
ask turtles [pendown forward 5]
```



Las tortugas tienen “tinta” para dibujar. cuando tienen el esférico sobre la superficie (pendown) el esférico es del mismo color que la tortuga.
Vamos a hacer ahora que cada tortuga dibuje un círculo a su alrededor

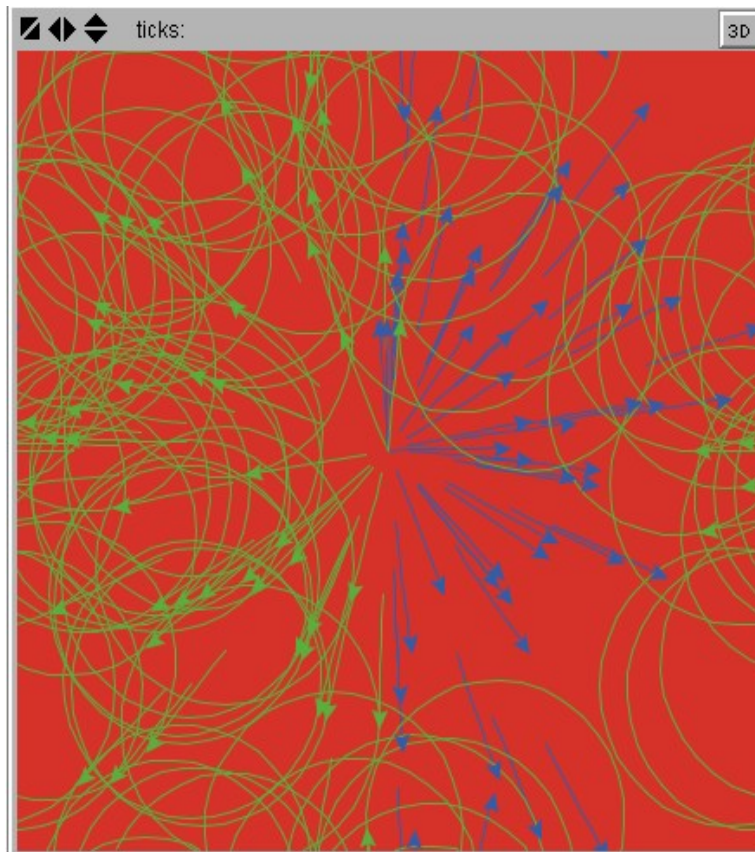
```
ask turtles [ if color = green [repeat 36 [forward 1 right 10]] ]
```



Los Parches

Adicionalmente a las tortugas , netLogo tiene otra clase de “agentes”,llos parches (patches). Estos son como las tortugas excepto que están siempre presentes y nunca se mueven. Cuando se inicializa la pantalla se crean estos agentes, o sea , a pesar de que la pantalla parezca vacía cuando no hay tortugas, los parches están ahí esperando comandos, coloquemos:

```
ask patches [set pcolor red]
```



Los parches se vuelven rojos, y si colocamos:

```
ask patches [if pxcor < 0 [set pcolor green]]
```



Listo puede en este momento seguir ensayando comandos en el próximo taller aprenderemos a usar estos comandos básicos para modificar modelos ya existentes de NetLogo.